

**steady state heat conduct**

学生姓名： 周麟

班 学 号： 111181

指导教师： 万林

**中国地质大学信息工程学院**

**2020年 11 月 19 日**

目录

[(一) **导言** 2](#_Toc56705225)

[**(二)** **问题的定义** 2](#_Toc56705226)

[问题描述 2](#_Toc56705227)

[迭代法 2](#_Toc56705228)

[**(三)** **拟议方法** 2](#_Toc56705229)

[Jacobi迭代法 2](#_Toc56705230)

[方法描述 2](#_Toc56705231)

[方法流程 3](#_Toc56705232)

[关键代码 3](#_Toc56705233)

[GaussSeidel迭代法 3](#_Toc56705234)

[方法描述 3](#_Toc56705235)

[方法流程 4](#_Toc56705236)

[关键代码 4](#_Toc56705237)

[SOR迭代法 5](#_Toc56705238)

[方法描述 5](#_Toc56705239)

[方法流程 6](#_Toc56705240)

[关键代码 6](#_Toc56705241)

[**a)** **方法比较** 6](#_Toc56705242)

[**i.** **Jacobi迭代与Gauss-Seidel迭代比较** 6](#_Toc56705243)

[**ii.** **Gauss-Seidel迭代与SOR迭代比较** 6](#_Toc56705244)

[**(四)** **实验** 6](#_Toc56705245)

[实验环境 6](#_Toc56705246)

[实验细节 7](#_Toc56705247)

[观察和结果分析 7](#_Toc56705248)

[结果分析 9](#_Toc56705249)

[**(五)** **结论** 9](#_Toc56705250)

[**(六)** **附录** 9](#_Toc56705251)

[Jacobi算法部分代码： 9](#_Toc56705252)

[Gauss- Seidel算法部分代码： 10](#_Toc56705253)

[SOR算法部分代码： 10](#_Toc56705254)

[参考文献： 11](#_Toc56705255)

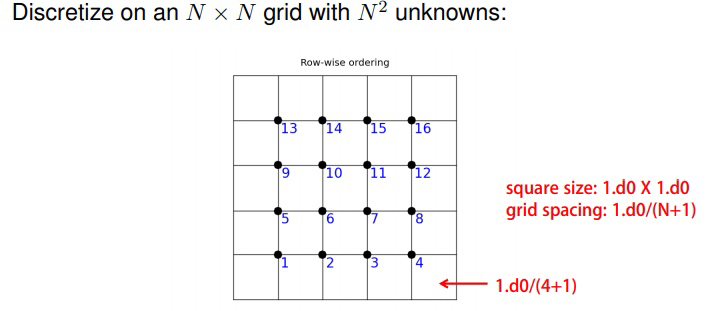
1. **导言**

本文探究热传导数值求解的问题，采用了不同计算方法，对其不同的计算方法的算法、具体实现进行描述。通过其实现结果来对不同计算方法进行时间、吞吐量和速度进行观察，得出其算法差异和运行的效率的快慢，分析其算法的优劣。

1. **问题的定义**

## 问题描述

本问题围绕着热传导问题的各点值问题进行求解，如图：



问题条件：

假设边界点的温度都已知，内部点的值近似为四个相邻值的平均值。即公式：

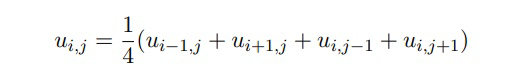


图1

i，j的值的范围都为：1，2，3...，N，则一共有N^2个未知数。

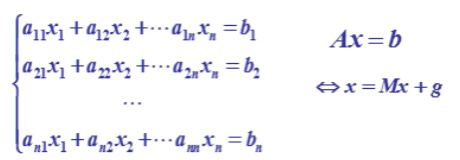
则可以假设为一个线性系统为一个矩阵A，矩阵A行列分布为N^2 X N^2 。

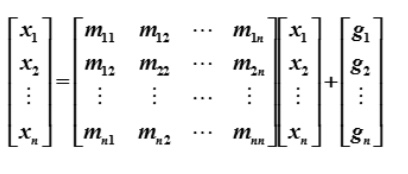
解出矩阵A的所有未知数的值。

因为只有边界点的值可知，所以一个未知点的附近四个相邻点的值不能完全已知；所以，求解的重要思想就是用化未知点为已知，通过迭代法，迭代至全部值已知，之后不断回溯，之后所有的点都已知，最后N^2个点的值全部解出。

## 迭代法

当我们求线性方程组的解时有以下关系：

****



将上述方程组变形可获得：

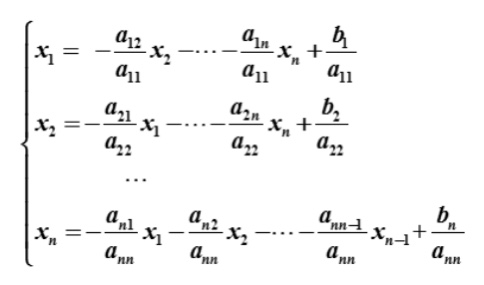


图2

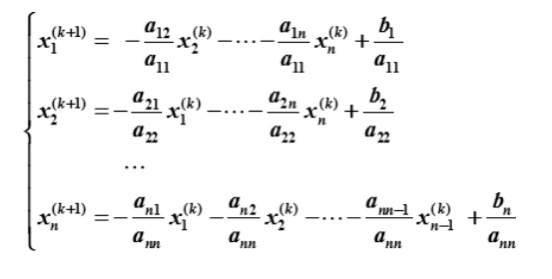
一般简单的式子可以手写计算使用待定系数法，但是对于计算机来说，计算数量庞大的方程式使用迭代法，类似大型稀疏矩阵（矩阵中有大部分元素都为0）时。

1. **拟议方法**

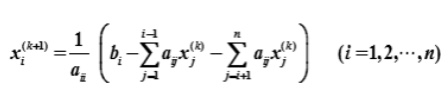
## Jacobi迭代法

### 方法描述

Jacobi迭代法是由图1式子变形得到：

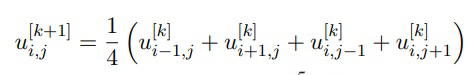


即：



下标i代表第i个值，下标k+1代表迭代到了k+1次。

本题线性方程由Jacobi迭代法表示为：



k+1代表第k+1次迭代，则可知第k+1次迭代是第k次迭代得到的结果进行表示的。

那么迭代的次数如何确定？

答：界定一个容差。每次迭代结束之后，检查所有变元的最大容差，用这个容差和设定容差进行比较，如果小于设定容差，结束迭代，直接输出结果。

### 方法流程

* 设定初始值。即：求解个数、容差、最大迭代数、线程数（为了测试不同线程数的效率） 并且设置输出值。
* 初始化x轴热源，y轴热源。（假设在x，y轴中假设长度为1，刻度0.3<= and >=0.7的温度为0，中间温度为1）初始化待求解温度矩阵，先假设每个未知点温度为0。
* 新建solution.txt文件。
* 双重循环中使用Jacobi迭代算法进行求值，小于设定容差结束迭代。
* 将数据写入solution.txt文件。

### 关键代码

2020-11-18 21:27:33.761000

设定线程私有变量

当线程并发进行时，发生顺序无法控制，这时候就需要进行线程私有操作，上述代码设定待求温度矩阵各点下标为私有，这样在进行并发操作时，不同线程下标互不干扰。

2020-11-18 21:13:05.688000

根据设定的线程数设置线程

2020-11-18 21:14:11.378000

根据Jacobi迭代方程计算相应的点的温度

2020-11-18 21:15:39.463000

每层迭代中的每一项都计算并更新所有变元的最大容差

2020-11-18 21:17:44.564000

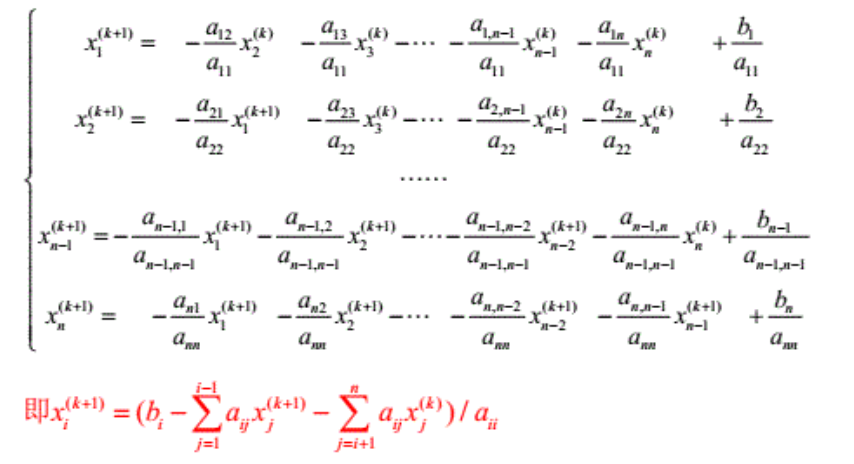
如果迭代中的最大容差小于设定容差，那么直接退出循环，输出数据

## GaussSeidel迭代法

### 方法描述

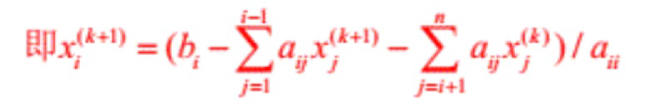
Gauss-seidel迭代法是对Jacobi迭代法的改进。研究Jacobi迭代法时，可以发现在逐个求X^(k+1)分量时，当计算到Xi^(k+1)时，分量X1^(k+1)，....，Xi-1^(k+1)都已经求得，而仍用旧分量Xi^(k),...，Xi-1^(k)计算Xi^(k+1)。

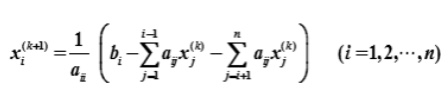
由于新算出的分量比旧分量准确些，因此设想一旦变量X1^(k+1)，...，Xi-1^(k+1)求出，马上就用新分量X1^(k+1)，...，Xi-1^(k+1)代替Jacobi迭代法中X1^(k)，...，Xi-1^(k) 来求Xi^(k+1)。



这样，每一个计算出来的分量Xi^(k+1)马上就可以用于计算下一个分量Xi+1^(k+1)，这种方法比Jacobi迭代法收敛得快。

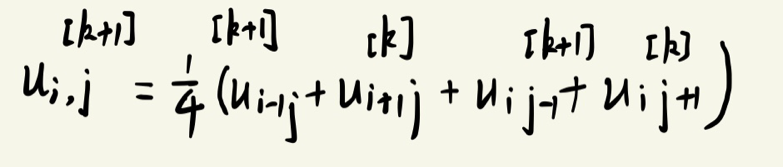
该式子与Jacobi迭代式不同的地方就是：





当求得任意一个迭代层数的未知量时，立刻用于求取同一迭代层数的未知量。

则在进行稳态热传导问题中，用Gauss-seidel迭代法表示的迭代式子为：



### 方法流程

方法流程和Jacobi方法相同，只不过算法不同。

### 关键代码

**2020-11-18 22:56:55.337000**

Causs-seidel迭代法算法

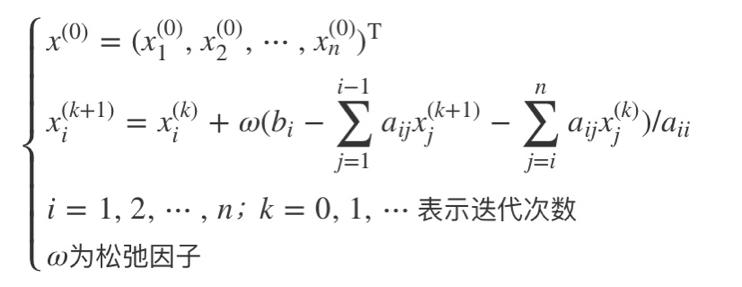
判断并更新当前迭代层数最大容差

## SOR迭代法

### 方法描述

SOR是Succesive Over Relaxation(逐次超松弛)的缩写。SOR迭代法是解大型稀疏矩阵方程组的有效方法之一。它可以看作是Gauss-seidel迭代法的加速，Gauss-seidel迭代法是SOR迭代的一种特殊形式。

逐次松弛SOR迭代法的计算公式为：



SOR迭代法当ω=1时, SOR迭代法即为Gauss-Seidel迭代法;当2>ω>1时, 称为超松弛迭代法;当ω<1时, 称为低松弛迭代法。

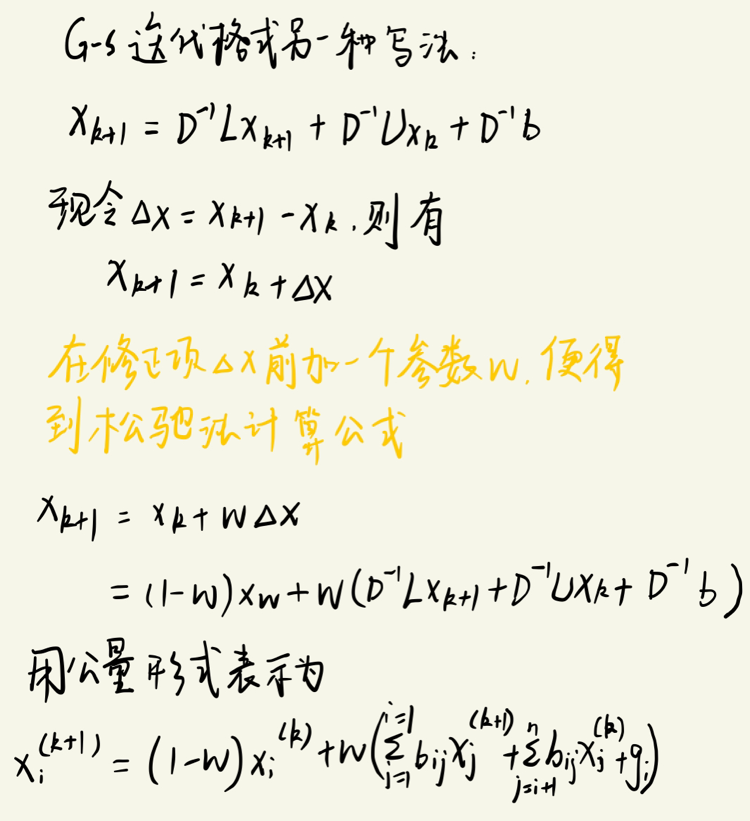
注2:松弛因子的选取对收敛速度影响极大, 但目前尚无可供实用的计算最佳松弛因子的方法，常常是根据系数矩阵的性质及实际经验，通过试算来确定较佳的松弛因子。

与Gauss- Seidel迭代法相比，通过引入外推参数w，可从Gauss- Seidel方法中推导出SOR。由于w的优化选择，SOR收敛比Gauss-seidel快很大一个量级。

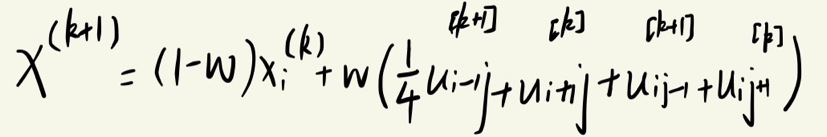
可以发现，当松弛因子为1时，变为Seidel迭代法，当调节松弛因子时，可以加快或减缓迭代速度。

Gauss- Seidel迭代式可以直接推导出SOR迭代式

推导过程为：



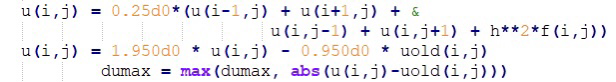
对于本题稳态热传导来说，用SOR式子表示为：



### 方法流程

同样地，与Jacobi迭代法运行流程相同，只不过算法部分不相同。

### 关键代码

****

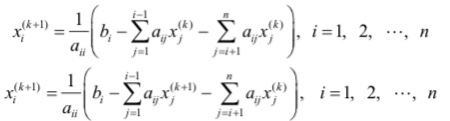
使用SOR迭代算法计算温度值，将松弛因子设为0.95

且计算和更新最大容差

* 1. **方法比较**

1. **Jacobi迭代与Gauss-Seidel迭代比较**

Jacobi和Gauss- Seidel的迭代形式分别为：



如果线性方程组A表示成A=D-L-U，其中D表示一个对角矩阵；L表示一个下三角矩阵，U表示一个上三角矩阵，则Jacobi迭代可用矩阵表示为：

2020-11-19 13:33:58.783000

相应地，Gauss- Seidel迭代可用矩阵表示为：

2020-11-19 13:34:49.219000

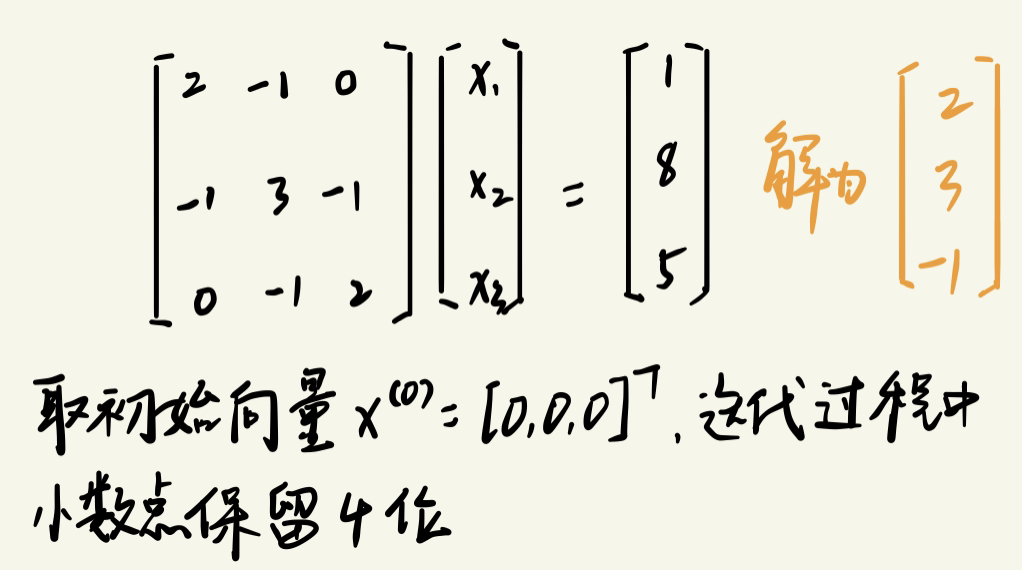
关于Jacobi迭代和Gauss- Seidel迭代的分析和比较，主要归结为：

* 收敛性。当且仅当各自的迭代矩阵（分别是Mj和Mg）的谱半径都小于1时，该迭代法收敛。若 A 是严格对角占优阵，则 2 个迭代法都收敛。但一般地， Jacobi 迭代收敛不能保证同一问题的 Gauss-Seidel迭代也收敛，反之亦然。
* 收敛速度。当 Gauss-Seidel 迭代和 Jacobi 迭代两者都收敛时，多数情况下Gauss-Seidel 迭代比Jacobi 迭代收敛得更快.对于一些模型问题，Gauss-Seidel 迭代的收敛速度甚至是 Jacobi 迭代的 2 倍以上。
* 每步计算量。矩阵D-L的求逆并不容易，所以具体用分量形式进行计算，即用一般的迭代形式来计算，两个迭代法每步的计算量相等。
* 存储量。Gauss- Seidel迭代只需要一个向量就可以存储x^(k)或x^(k+1)分量，而Jacobi需要两个向量分别存储。

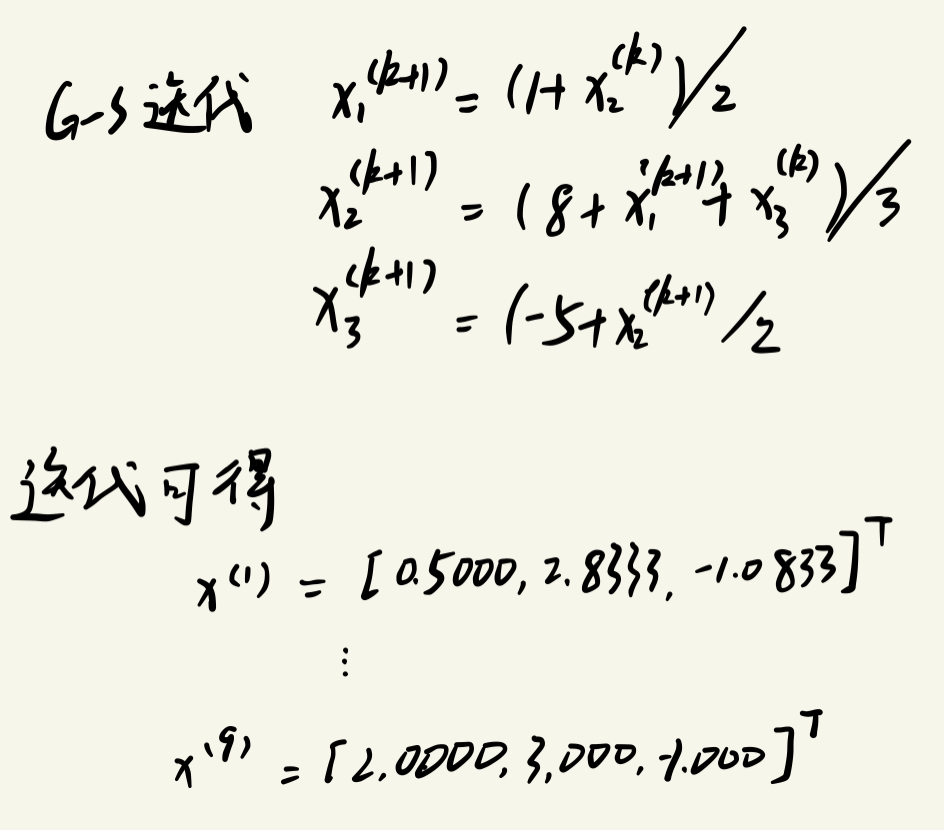
比较可知，在传统算法实现中，Gauss- Seidel迭代比Jacobi迭代往往更具优势。

1. **Gauss-Seidel迭代与SOR迭代比较**

分别用Gauss- Seidel，SOR迭代解线性方程组：

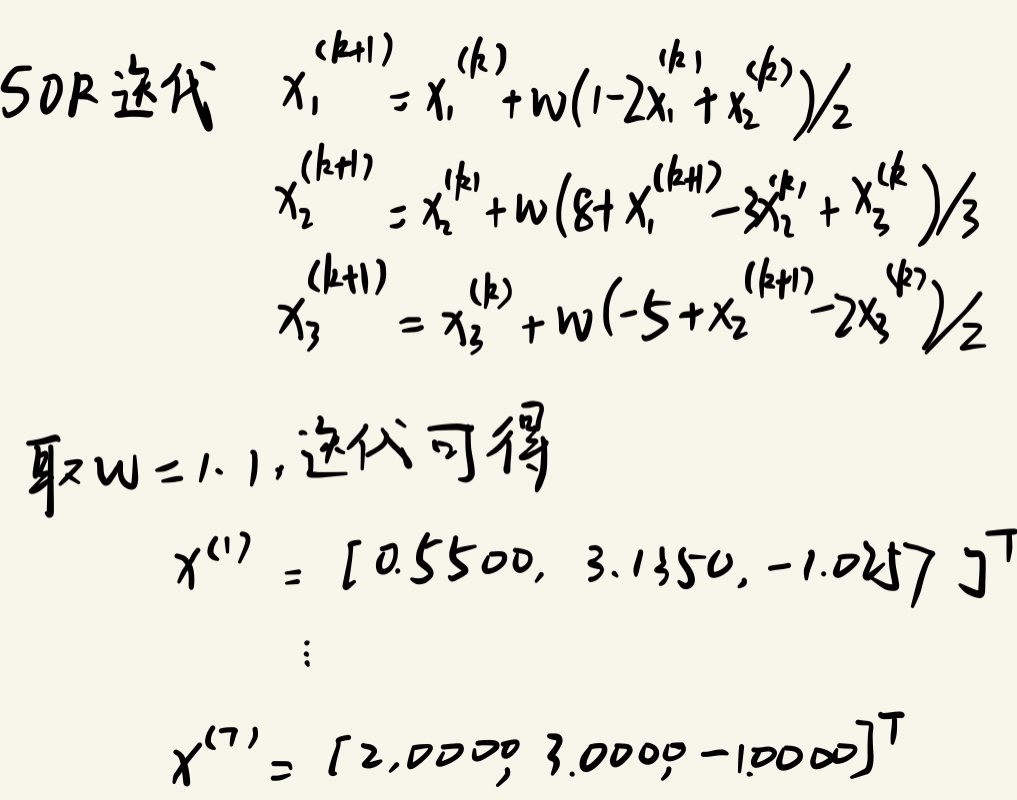


用Gauss- Seidel迭代法求解得



一共迭代了9次

用SOR迭代法求解得：



一共迭代了7次

由结果得知，显然SOR迭代法收敛速度比Gauss-Seidel迭代效率高，但是在实际运用中，确定最优松弛因子却是非常困难的。

1. **实验**

## 实验环境

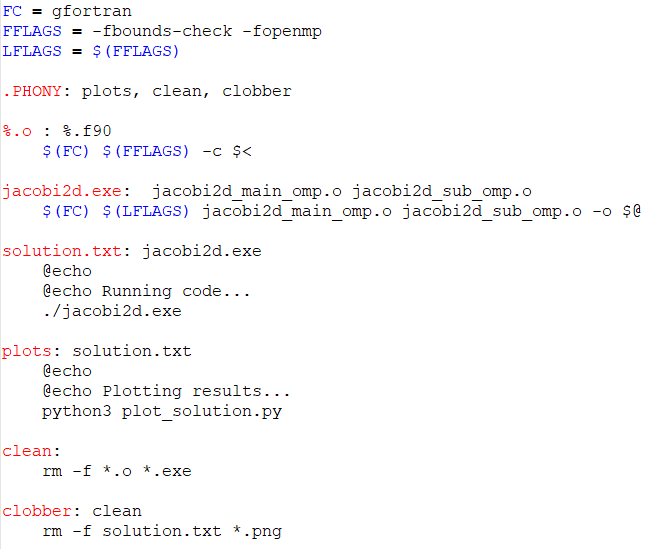
本问题是在Xubuntu中运行的，先使用Fortran进行迭代算法编码，在用python进行结果写入文件。最后通过编写makefile在命令行中运行程序。

## 实验细节

在进行试验中，最重要的步骤就是编写makefile文件。

使用makefile可以把多个文件进行编译和链接；这样，在运行时只需要一条指令就可以完成全部工作。

makefile内容：



.PHONY代表命令行可以make的操作 这里有plots，clean，clobber

* 显然clean操作就是将生成的可执行文件.exe,目标文件.o删除
* clobber操作是将计算出的结果文件和图片文件删除
* plot则是链接目标文件后生成可执行文件，运行可执行文件，生成结果文件，将结果文件输入python代码中读取数据，然后将数据绘制进行可视化，保存至.png文件中。

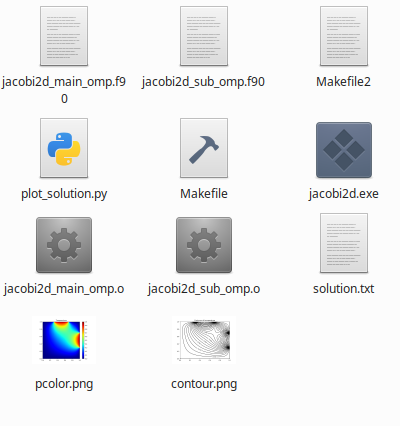
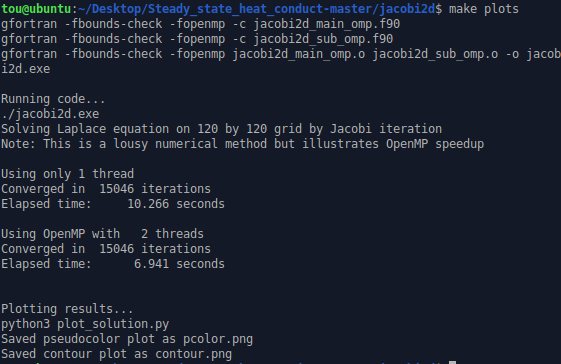
运行方法：

在存放不同迭代算法代码路径中打开终端，输入make plots，即可运行程序。

## 观察和结果分析

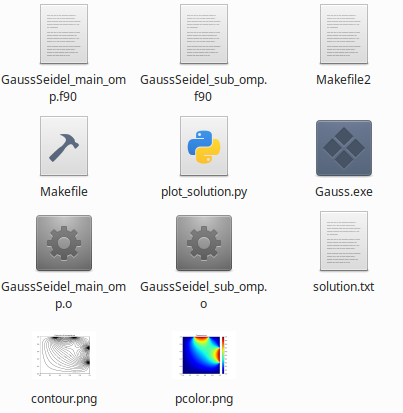
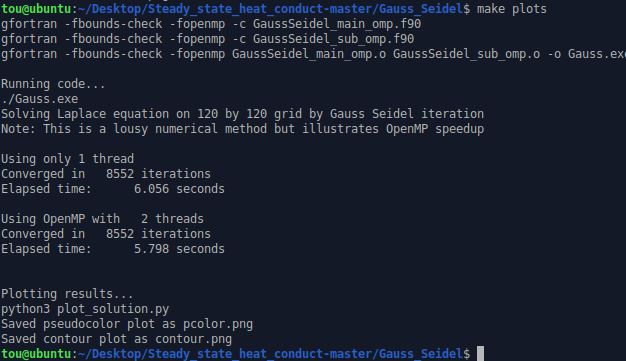
运行Jacobi迭代

运行结果：



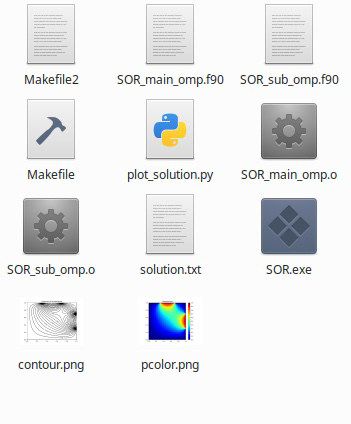
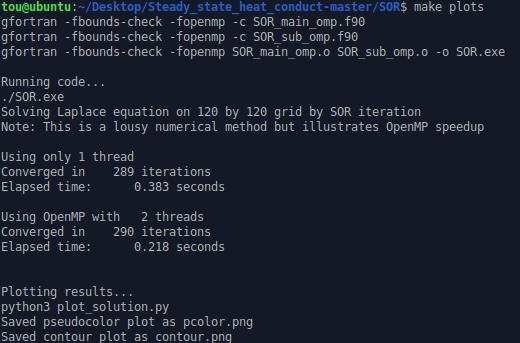
运行Gauss-Seidel迭代

运行结果：



运行SOR迭代

运行结果：



## 结果分析

可以发现，每种方法之要进行多线程运行，计算效率就会提高。

对于Jacobi、Gauss-Seidel、SOR迭代进行迭代次数和用时分析

在单线程中：

* 迭代次数

Jacobi迭代

15046次

Gauss- Seidel迭代

8552次

SOR迭代

289次

可以发现 Jacobi收敛速度最低，而在Gauss-Seidel迭代基础上加了松弛因子的SOR迭代效率最高。

* 总用时

Jacobi迭代

10.266s

Gauss- Seidel迭代

6.056s

SOR迭代

0.383s

可以发现SOR迭代的用时是Jacobi迭代的30多倍，是Gauss-Seidel迭代的20多倍，SOR迭代效率显然远远高于Jacobi，和Gauss-Seidel迭代。

1. **结论**

通过这次课程的学习，了解了三种线性方程进行迭代求解的算法，通过查阅了若干的相关材料，对Jacobi，Gauss-Seidel，SOR迭代法之间的相互联系，算法有了一定的了解。对于Fortran语言进行了简单的了解，在终端通过makefile对整个项目目标文件进行链接运行，对虚拟机的操作进行了进一步的熟悉和了解。

通过对“稳态热传导”这个问题的研究发现，这三种方法中，Gauss-Seidel的效率虽然高于Jacobi，但是没有显著的增加，而对于SOR迭代则是明显效率远远高于前两种迭代效率，做出一个假设，选择其他的松弛因子，可能效率会更高。

1. **附录**

## Jacobi算法部分代码：

do iter=1,maxiter

uold = u ! old values

dumax = 0.d0

!$omp parallel do private(i) reduction(max : dumax)

! private - only be accessed by a single thread.

do j=1,n

do i=1,n

u(i,j) = 0.25d0\*(uold(i-1,j) + uold(i+1,j) + &

uold(i,j-1) + uold(i,j+1) + h\*\*2\*f(i,j))

dumax = max(dumax, abs(u(i,j)-uold(i,j)))

enddo

enddo

if (mod(iter,nprint)==0) then

print 203, iter, dumax

203 format("After ",i8," iterations, dumax = ",d16.6,/)

endif

! check for convergence:

! 误差小于

if (dumax .lt. tol) exit

enddo

## Gauss- Seidel算法部分代码：

do iter=1,maxiter

uold = u ! old values

dumax = 0.d0

! $omp parallel do private(i) reduction(max : dumax)

! private - only be accessed by a single thread.

do j=1,n

do i=1,n

u(i,j) = 0.25d0\*(u(i-1,j) + uold(i+1,j) + &

u(i,j-1) + uold(i,j+1) + h\*\*2\*f(i,j))

dumax = max(dumax, abs(u(i,j)-uold(i,j)))

enddo

enddo

if (mod(iter,nprint)==0) then

print 203, iter, dumax

203 format("After ",i8," iterations, dumax = ",d16.6,/)

endif

! check for convergence:

if (dumax .lt. tol) exit

enddo

## SOR算法部分代码：

do iter=1,maxiter

uold = u ! old values

dumax = 0.d0

!$omp parallel do private(i) reduction(max : dumax)

! private - only be accessed by a single thread.

do j=1,n

do i=1,n

u(i,j) = 0.25d0\*(u(i-1,j) + u(i+1,j) + &

u(i,j-1) + u(i,j+1) + h\*\*2\*f(i,j))

u(i,j) = 1.950d0 \* u(i,j) - 0.950d0 \* uold(i,j)

dumax = max(dumax, abs(u(i,j)-uold(i,j)))

enddo

enddo

if (mod(iter,nprint)==0) then

print 203, iter, dumax

203 format("After ",i8," iterations, dumax = ",d16.6,/)

endif

! check for convergence:

! 误差小于

if (dumax .lt. tol) exit

enddo

# 参考文献：

[1]吴丹宇.Jacobi迭代与Gauss-Seidel迭代的比较[R].黄冈师范学院,数学系,湖北,黄冈:仲恺农业技术学院学报,2005, 18(3).

[2]李焕荣.浅谈线性方程组的迭代求解[J].重庆工商大学学报(自然科版),2012,07:28-32.